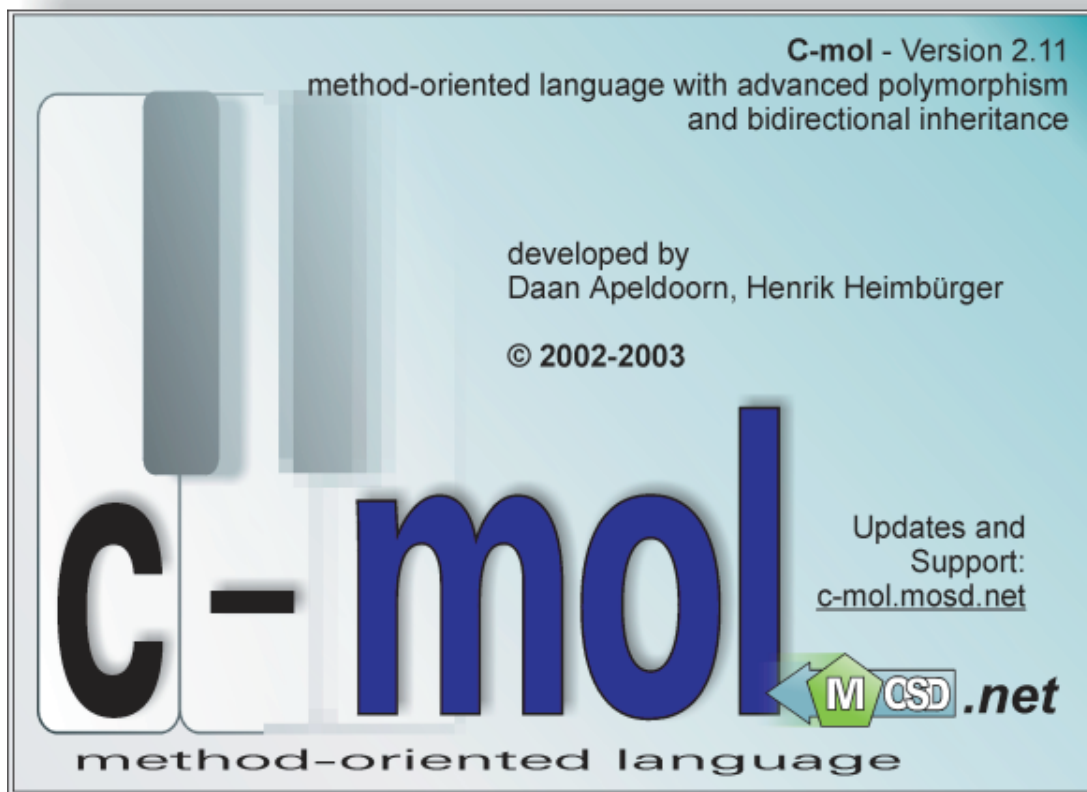


Daan Apeldoorn • Henrik Heimbürger



## Das Handbuch

### Entpacken

Laden Sie sich eines der Compiler-Archive herunter und entpacken Sie es. Die Windows-Version ist ein Installationsprogramm. Die Linux-Version liegt in Form der dort üblichen gzip-tar-Kombination vor. Das Zielverzeichnis ist prinzipiell beliebig. Wir empfehlen z.B. "c:\Programme\C-mol-Compiler" für Windows und "/usr/bin/cmc" für Linux.

Wird der Compiler aus der Kommandozeile heraus gestartet - d.h. nicht aus einer IDE wie z.B. dem Microsoft Visual Studio - so sollte der Compiler in den Systempfad eingetragen werden. Dies ist z.B. unter Dos in der "autoexec.bat", unter Windows in der Systemsteuerung und unter Linux in dem jeweiligen Startup-Skript (je nach Shell z.B. mit dem Befehl "setenv") möglich.

### Dateien

Der Compiler besteht in erster Linie aus den in diesem Archiv enthaltenen .py-Dateien. Dies sind Quellcodes in der Sprache Python. Gestartet wird immer nur die "c-mol.py", alle anderen Dateien werden dann von dieser geladen und aufgerufen.

Darüber hinaus enthält das Archiv diverse Konfigurationsdateien (config.\*) und das Konfigurationstool (Config.jar). Informationen über diese Dateien und die Verwendung des Tools entnehmen Sie bitte [Handbuch: Konfigurationsprogramm](#).

Weiterhin sind in der Windows-Version die Datei "SplashScreen.exe" und das *C-mol*-Logo in visueller und auditiver Form enthalten. Auch diese werden selbstständig vom Compiler aufgerufen.

## Konfigurationsprogramm

Mit dem Konfigurationsprogramm ist es möglich, den *C-mol*-Compiler schnell auf den Microsoft Visual C++- bzw. den GCC-Compiler umzustellen oder ihn sogar auf andere Compiler zu konfigurieren.

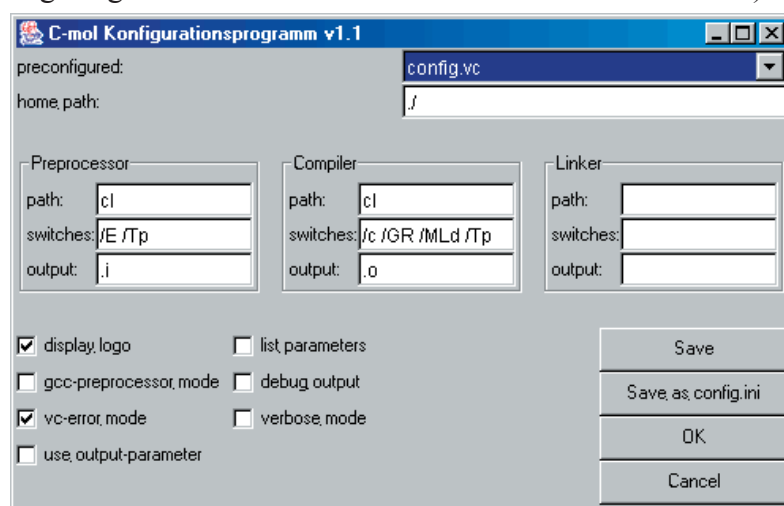
Damit dabei auch die Plattformunabhängigkeit erhalten bleibt, wurde das Konfigurationsprogramm in Java entwickelt. Herunterladen können Sie es unter [Downloads](#).

Zum Starten ist eine erfolgreich installierte Java-Virtual-Machine erforderlich. Entwickelt wurde das Konfigurationsprogramm unter Java 1.40, sollte jedoch auch mit niedrigeren Version lauffähig sein. Die erforderlichen Befehle sind von der Plattform und Virtual Machine abhängig. Z.B. bei einer unter Windows 2000 installierten Java 1.4-VM, die auch im Path eingetragen ist, ist dies in der Console mit der Befehlszeile

```
javaw -jar Config.jar
```

möglich.

Dann sollte ein Fenster ähnlich dem folgenden (es sollte normalerweise im Stil ihres Betriebssystems angezeigt werden - hier zu sehen ist die Windows-Version) erscheinen:



Normalerweise sollte eine manuelle Konfiguration nur nötig sein, wenn der *C-mol*-Compiler nicht mit den standardmäßig unterstützten C++-Compilern verwendet wird. In diesem Fall reicht es also, je nach verwendetem Compiler unter “preconfigured” den Visual C++ oder den GCC auszuwählen und dann auf “Save as config.ini” zu klicken.

Trotzdem kann es in manchen Situationen natürlich sinnvoll sein, den *C-mol*-Compiler an die eigenen Ansprüche anzupassen bzw. ihn auf neue Compiler einzustellen. Dafür bietet das Konfigurationsprogramm folgende Optionen:

**preconfigured:** Aus dieser ComboBox können die drei vordefinierten Konfigurationsdateien ausgewählt werden. In einer späteren Version werden hier sicherlich auch eigene Dateien erstellbar sein. Zu beachten ist, dass der *C-mol*-Compiler grundsätzlich die "config.ini" verwendet. Die "config.vc" ist nur eine Vordefinition für den Microsoft Visual C++-Compiler und die "config.gcc" eine Vordefinition für den GNU C++-Compiler.

**home path:** Hier kann der Pfad des *C-mol*-Compilers angegeben werden. Wird hier kein Pfad angegeben (bzw. "./"), so sollte der Aufruf immer aus dem Verzeichnis des *C-mol*-Compilers erfolgen.

**Preprocessor / Compiler / Linker:** Für die drei externen Module Preprocessor, Compiler und Linker können jeweils die folgenden Optionen eingestellt werden:

**path:** Hier muss Pfad und Dateiname des Moduls angegeben werden. Wird kein Pfad angegeben, so muss sich das Modul im Systempfad befinden. Wird hier gar nichts angegeben, so wird das entsprechende Modul nicht gestartet (so z.B. der Fall beim Linker für den Microsoft Visual C++-Compiler, da der Linker vom Visual Studio eigenhändig aufgerufen wird).

**switches:** Ggf. benötigte Parameter, die beim Aufruf des Moduls über die Kommandozeile übergeben werden sollen.

**output:** Hier muss angegeben werden, welche Dateierweiterung die durch dieses Modul entstehenden Dateien besitzen, damit der *C-mol*-Compiler weiß, womit er weiterarbeiten kann.

**display logo:** Soll beim Compilieren der *C-mol*-Splashscreen angezeigt werden (nur unter Windows)?

**list parameters:** Sollen alle vom *C-mol*-Compiler verwendeten Optionen aufgelistet werden?

**gcc-preprocessor mode:** Der GCC-Preprocessor erzeugt etwas anderen Code als der Visual C++-Preprocessor. Hiermit kann eingestellt werden, welcher Code erwartet werden kann.

**use output-parameter:** Der GCC bietet die spezielle "-o"-Option, um den Dateinamen der zu erstellenden ausführbaren Datei anzugeben. Soll diese Option verwendet werden?

**verbose mode:** Wenn diese Option aktiviert ist, berichtet der *C-mol*-Compiler über jeden Arbeitsschritt den er gerade durchführt. Dies ist normalerweise nur bei Installationsproblemen des Compilers notwendig.

**debug mode:** Aktiviert weitere Ausgaben des Compilers, die nur bei Problemen innerhalb des Compilers notwendig sein dürften.

**vc-error mode:** Bei Aktivierung gibt der *C-mol*-Compiler die Fehlermeldungen im Stil des Visual C++-Compilers aus, so dass z.B. im Visual Studio direkt mit Doppelklick zu dem Fehler gesprungen werden kann. Bei nicht-Aktivierung gibt er die Fehler im Stil des GCC aus, so dass die gleiche Funktion z.B. im Falch.net Developer Studio verwendet werden kann.

**Save:** Speichert alle Optionen in der momentan unter "preconfigured" gewählten Datei.

**Save as config.ini:** Speichert alle Optionen in der Datei "config.ini". Damit ist ein schnelles Umschalten z.B. zwischen GCC und Visual C++ möglich. Es wird einfach unter "preconfigured" die "config.vc" oder "config.gcc" ausgewählt und dann auf diesen Button geklickt. Schon werden die entsprechenden Optionen für den nächsten Start des *C-mol*-Compilers übernommen.

**OK:** Speichert alle Optionen in der momentan unter "preconfigured" gewählten Datei und beendet das Konfigurationsprogramm.

**Cancel:** Beendet das Konfigurationsprogramm ohne irgend etwas zu speichern.

## Verwendung des Compilers im Microsoft Visual Studio

Zur Verwendung des Compilers im Visual Studio sind zwei Änderungen vorzunehmen. Zum einen muss in den Optionen der Pfad zum *C-mol*-Compiler angegeben werden. Dies lässt sich im Menü unter

```
Tools
  --> Options
    --> Directories
      --> Show directories for: Executable files
```

einstellen. Dort muss der exakte Pfad zum *C-mol*-Compiler hinzugefügt werden. Ein einfacher Eintrag im Systempfad reicht für das Visual Studio nicht aus, da es diesen nicht übernimmt! Weiterhin ist für jede *C-mol*-Datei ein Custom Build Step einzurichten. Dies ist leider für jede Datei einzeln notwendig. Dazu sind folgende Schritte erforderlich:

```
im Workspace
  --> im Kontextmenü der C-mol-Datei (Mausklick rechts)
    --> Settings
      --> unter General
        --> Always use custom build step aktivieren
      --> und unter Custom Build
        --> Description:    Compiling C-mol...
        --> Commands:     c-mol.py $(InputPath)
        --> Outputs:      $(InputDir)\$(InputName).obj
```

angeben.

Weiterhin sollte wie unter [Handbuch: Konfigurationsprogramm](#) beschrieben, die Konfigurationsdatei für den Microsoft Visual C++-Compiler aktiviert werden.

Zu guter Letzt ist es noch möglich, das Syntaxhighlighting für die Datei zu aktivieren. Dafür ist im mit der rechten Maustaste auf das geöffnete Editorfenster zu klicken und dort "Properties" auszuwählen. Dort lässt sich unter "Language" "C/C++" auswählen. Die methodenorientierten Erweiterungen fehlen natürlich, aber es ist ein Anfang...

## Verwendung der Sprache

Nun steht dem Einsatz von *C-mol* nichts mehr im Wege. Sie können nun die methodenorientierten Statements, Deklarationen und Definitionen gemäß der *C-mol*-Syntax verwenden und natürlich auch mit strukturierten oder objektorientierten Konzepten vermischen.

Möchten Sie einen weiteren Einstieg in die methodenorientierte Programmierung erhalten, empfehlen wir Ihnen nun, sich von der [Homepage](#) die vollständige Dokumentation herunterzuladen und zumindest die Kapitel 1 und 2 zu lesen. So bekommen Sie u.a. eine exakte Beschreibung der Möglichkeiten der methodenorientierten Softwareentwicklung und der Syntax von *C-mol*. Weiterhin können Sie sich dort einige Demoprogramme herunterladen, ausprobieren und etwas damit herumspielen. So bekommen Sie einen sehr guten Eindruck davon, welche Möglichkeiten *C-mol* bietet.

### Headerstop-Pragma

Eine weitere Besonderheit existiert noch. Um die Geschwindigkeit des Compilervorgangs zu beschleunigen, können sie eine spezielle Pragma (eine Anweisung an den Compiler) verwenden, um die rein strukturierten oder objektorientierten Header-Dateien von der Suche nach methodenorientierten Konzepten auszuschließen. Dazu verwenden Sie bitte den Befehl

```
#pragma c-mol_hdrstop
```

vor der ersten methodenorientierten Header-Datei oder dem ersten methodenorientierten Befehl. Alles bis zu dieser Direktive wird dann zwar von dem C++-Compiler, nicht jedoch von dem *C-mol*-Compiler verarbeitet. Natürlich können Sie auch weiterhin diese Direktive weglassen, dann wird eben jede Datei methodenorientiert verarbeitet.